Facebook Hacker Cup 2018 Round 1

Request Clarification

Platform Parkour

28 points

Download Input

You're about to put on an exciting show at your local circus — a parkour demonstration! **N** platforms with adjustable heights have been set up in a row, and are numbered from 1 to **N** in order from left to right. The initial height of platform *i* is H_i metres.

When the show starts, **M** parkourists will take the stage. The *i*th parkourist will start at platform A_i, with the goal of reaching a different platform B_i. If $B_i > A_i$, they'll repeatedly jump to the next platform to their right until they reach B_i. If $B_i < A_i$, they'll instead repeatedly jump to the next platform to their left until they reach B_i. All of the parkourists will complete their routes simultaneously (but don't worry, they've been trained well to not impede one another).

Not all parkourists are equally talented, and there are limits on how far up or down they can jump between successive platforms. The *i*th parkourist's maximum upwards and downwards jump heights are U_i and D_i, respectively. This means that they're only able to move directly from platform *x* to some adjacent platform *y* if $H_x - D_i \le H_y \le H_x + U_i$, where H_x and H_y are the current heights of platforms *x* and *y*, respectively.

With the show about to begin, a disastrous flaw has just occurred to you — it may not be possible for all of the parkourists to actually complete their routes with the existing arrangement of platforms! If so, you will need to quickly adjust some of the platforms' heights first. The height of each platform may be adjusted upwards or downwards at a rate of 1 metre per second, to any non-negative real-valued height of your choice, and multiple platforms may be adjusted simultaneously. As such, if the initial height of platform *i* is H_i and its final height is P_i, then the total time required to make your chosen height adjustments will be max{$|H_i - P_i|$} over *i*=1..**N**.

Determine the minimum amount of time required to set up the platforms such that all **M** parkourists will then be able to complete their required routes. Note that you may not perform further height adjustments once the show starts. The platform heights must all remain constant while all **M** parkourists complete their routes.

In order to reduce the size of the input data, you're given H_1 and H_2. $H_{3..N}$ may then be generated as follows using given constants **W**, **X**, **Y**, and **Z** (please watch out for integer overflow during this process):

$H_i = (W * H_{i-2} + X * H_{i-1} + Y) \% Z$ (for *i*=3..**N**)

Input

Input begins with an integer **T**, the number of shows. For each show, there is first a line containing the space-separated integers **N** and **M**. The next line contains the space-separated integers H_1, H_2, **W**, **X**, **Y**, and **Z**. Then, **M** lines follow. The *i*th of these lines contains the space-separated integers A_i, B_i, U_i, and D_i.

Output

For the *i*th show, print a line containing "Case #*i*: " followed by 1 real number, the minimum amount of time required to set up the platforms (in seconds). Absolute and relative errors of up to 10^{-6} will be ignored.

Constraints

$1 \le T \le 85$
$2 \le N \le 200{,}000$
$1 \le M \le 20$
$0 \le H_i < Z$
$0 \le W, X, Y < Z$
$1 \le Z \le 1{,}000{,}000$
$1 \le A_i, B_i \le N$
$0 \le U_i, D_i \le 1{,}000{,}000$
$A_i, \ne B_i$

Explanation of Sample

In the first case, **H** = [0, 10]. You can increase the first platform's height by 3.5 and decrease the second's by 3.5 in 3.5 seconds, yielding **P** = [3.5, 6.5]. The single parkourist will then be able to successfully complete their route from platform 1 to platform 2 by jumping upwards by a height of at most 3.

In the second case, **H** = [50, 59, 55, 51, 47]. One optimal possibility is **P** = [54.0, 54.5, 53.5, 52.5, 51.5].

In the third case, **H** = [46, 38, 38, 22, 8].

In the fourth case, **H** = [53, 25, 24, 81, 77, 40, 29, 21].

Sample input · Download

```
5
2 1
0 10 0 0 0 11
1 2 3 8
5 1
50 59 0 1 96 100
1 5 2 1
5 2
46 38 2 4 44 50
```

Sample output · Download

```
Case #1: 3.500000
Case #2: 4.500000
Case #3: 15.000000
Case #4: 24.000000
Case #5: 483009.500000
```

```
3 5 4 1
8 5
53 25 15 23 54 100
1 8 14 9
3 1 5 7
6 8 2 1
5 4 1 8
8 1 11 10
100000 5
72464 815932 291056 735002 4758 972844
68327 29055 2880 3051
98105 26231 3531 3141
4018 31397 2797 3619
92594 65725 3824 3003
81932 8087 3372 3158
```